EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	0	((cache same ((block near2 code) near count$2)) and evict$4).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/17 13:30
L2	0	((cache same ((block near2 code) with count$2)) and evict$4).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/17 13:31
L3	0	((cache and ((block near2 code) with count$2)) and evict$4).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/17 13:31
L4	0	((cache and ((block near2 code) same count$2)) and evict$4).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/17 13:31
L5	3	((cache and ((block) same count$2)) and evict$4).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/17 13:43
L6	0	((cache and ((block) same count$2)) and evict$4).ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/17 13:43
L7	0	((cache and ((block) same count$2)) and (dynamic$5 same (optimi$4 management LRU))).ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/17 13:44
L8	11	((cache same count$2) and (dynamic$5 same (optimi$4 management LRU))).ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/17 13:45
L9	2	8 and @rlad<"20010703"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/17 13:51

EAST Search History

L10	2	9 and (counter near8 cach$3)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/17 13:49
L11	0	8 and ((add$3 instrument$4 insert$4 inlin$3) same (code and block))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/17 13:50
L12	96	(counter near4 cache) and (LRU eviction) and ((add$3 instrument$4 insert$4 inlin$3) same (code and block))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/17 13:51
L13	42	12 and @rlad<"20010703" and (((instruction code) and counter) near4 (cache cached caching))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/17 13:53
L14	0	13 and ((block near2 code) same ((tracking counter increment$5) and cache))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/17 13:54

Terms used **counter cache** and **LRU** and **code cache** and **code insertion near9 incrementing counter**

Found **100,416** of **198,617**

Sort results by relevance
Display results expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Generational Cache Management of Code Traces in Dynamic Optimization Systems
Kim Hazelwood, Michael D. Smith
December 2003 **Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture MICRO 36**
Publisher: IEEE Computer Society
Full text available: pdf(393.80 KB) Additional Information: full citation, abstract, citings, index terms

A dynamic optimizer is a runtime software system thatgroups a program's instruction sequences into traces, optimizesthose traces, stores the optimized traces in a software-basedcode cache, and then executes the optimized code inthe code cache. To maximize performance, the vast majorityof the program's execution should occur in the codecache and not in the different aspects of the dynamic optimizationsystem. In the past, designers of dynamic optimizershave used the SPEC2000 benchmark suite to jus ...

2 The design, implementation, and evaluation of adaptive code unloading for resource-constrained devices
Lingli Zhang, Chandra Krintz
June 2005 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 2 Issue 2
Publisher: ACM Press
Full text available: pdf(814.17 KB) Additional Information: full citation, abstract, references, index terms

Java Virtual Machines (JVMs) for resource-constrained devices, e.g., hand-helds and cell phones, commonly employ interpretation for program translation. However, compilers are able to produce significantly better code quality, and, hence, use device resources more efficiently than interpreters, since compilers can consider large sections of code concurrently and exploit optimization opportunities. Moreover, compilation-based systems store code for reuse by future invocations obviating the redund ...

Keywords: Code unloading, JIT, JVM, code-size reduction, resource-constrained devices

3 Instrumentation: Framework for instruction-level tracing and analysis of program executions
Sanjay Bhansali, Wen-Ke Chen, Stuart de Jong, Andrew Edwards, Ron Murray, Milenko Drinić, Darek Mihočka, Joe Chau
June 2006 **Proceedings of the second international conference on Virtual execution**

environments VEE '06
Publisher: ACM Press
Full text available: pdf(227.87 KB) Additional Information: full citation, abstract, references, index terms

Program execution traces provide the most intimate details of a program's dynamic behavior. They can be used for program optimization, failure diagnosis, collecting software metrics like coverage, test prioritization, etc. Two major obstacles to exploiting the full potential of information they provide are: (*i*) performance overhead while collecting traces, and (*ii*) significant size of traces even for short execution scenarios. Reducing information output in an execution trace can r ...

Keywords: callback, code emulation, code replay, time-travel debugging, tracing

4 Work in progress session: tools: Detailed cache simulation for detecting bottleneck, miss reason and optimization potentialities
Jie Tao, Wolfgang Karl
October 2006 **Proceedings of the 1st international conference on Performance evaluation methodolgies and tools valuetools '06**
Publisher: ACM Press
Full text available: pdf(184.39 KB) Additional Information: full citation, abstract, references, index terms

Cache locality optimization is an efficient way for reducing the idle time of modern processors in waiting for needed data. This kind of optimization can be achieved either on the side of programmers or compilers with code level optimization or at system level through appropriate schemes, like reconfigurable cache organization and adequate prefetching or replacement strategies. For the former users need to know the problem, the reason, and the solution, while for the latter a platform is require ...

Keywords: cache simulation, code optimization, performance visualization

5 Cache decay: exploiting generational behavior to reduce cache leakage power
Stefanos Kaxiras, Zhigang Hu, Margaret Martonosi
May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture ISCA '01**, Volume 29 Issue 2
Publisher: ACM Press
Full text available: pdf(1.17 MB) Additional Information: full citation, abstract, references, citings, index terms

Power dissipation is increasingly important in CPUs ranging from those intended for mobile use, all the way up to high-performance processors for high-end servers. While the bulk of the power dissipated is dynamic switching power, leakage power is also beginning to be a concern. Chipmakers expect that in future chip generations, leakage's proportion of total chip power will increase significantly.

This paper examines methods for reducing leakage power within the cache memori ...

6 Security and correctness: Efficient data protection for distributed shared memory multiprocessors
Brian Rogers, Milos Prvulovic, Yan Solihin
September 2006 **Proceedings of the 15th international conference on Parallel architectures and compilation techniques PACT '06**
Publisher: ACM Press
Full text available: pdf(386.29 KB) Additional Information: full citation, abstract, references, index terms

Data security in computer systems has recently become an increasing concern, and

hardware-based attacks have emerged. As a result, researchers have investigated hardware encryption and authentication mechanisms as a means of addressing this security concern. Unfortunately, no such techniques have been investigated for Distributed Shared Memory (DSM) multiprocessors, and previously proposed techniques for uni-processor and Symmetric Multiprocessor (SMP) systems cannot be directly used for DSMs. T ...

Keywords: DSM multiprocessor, data security, memory encryption and authentication

7 The measured performance of personal computer operating systems

J. B. Chen, Y. Endo, K. Chan, D. Mazieres, A. Dias, M. Seltzer, M. D. Smith

December 1995 **ACM SIGOPS Operating Systems Review , Proceedings of the fifteenth ACM symposium on Operating systems principles SOSP '95**, Volume 29 Issue 5

Publisher: ACM Press

Full text available: pdf(1.98 MB) Additional Information: full citation, references, citings, index terms

8 Compact Binaries with Code Compression in a Software Dynamic Translator

Stacey Shogan, Bruce R. Childers

February 2004 **Proceedings of the conference on Design, automation and test in Europe - Volume 2 DATE '04**

Publisher: IEEE Computer Society

Full text available: pdf(102.73 KB) Additional Information: full citation, abstract, citings, index terms

Embedded software is becoming more flexible and adaptable, which presents new challenges for management of highly constrained system resources. Software dynamic translation (SDT) has been used to enable software malleability at the instruction level for dynamic code optimizers, security checkers, and binary translators. This paper studies the feasibility of using SDT to manage program code storage in embedded systems. We explore to what extent code compression can be incorporated in a software i ...

9 Improving Region Selection in Dynamic Optimization Systems

David Hiniker, Kim Hazelwood, Michael D. Smith

November 2005 **Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture MICRO 38**

Publisher: IEEE Computer Society

Full text available: pdf(432.86 KB) Publisher Site Additional Information: full citation, abstract, citings, index terms

The performance of a dynamic optimization system depends heavily on the code it selects to optimize. Many current systems follow the design of HP Dynamo and select a single interprocedural path, or trace, as the unit of code optimization and code caching. Though this approach to region selection has worked well in practice, we show that it is possible to adapt this basic approach to produce regions with greater locality, less needless code duplication, and fewer profiling counters. In particular ...

10 Dynamo: a transparent dynamic optimization system

Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(156.03 KB) Additional Information: full citation, abstract, references, citings, index terms

We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the ...

11 Managing bounded code caches in dynamic binary optimization systems

Kim Hazelwood, Michael D. Smith
September 2006 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 3 Issue 3
Publisher: ACM Press
Full text available: pdf(666.72 KB) Additional Information: full citation, abstract, references, index terms

Dynamic binary optimizers store altered copies of original program instructions in software-managed code caches in order to maximize reuse of transformed code. Code caches store code blocks that may vary in size, reference other code blocks, and carry a high replacement overhead. These unique constraints reduce the effectiveness of conventional cache management policies. Our work directly addresses these unique constraints and presents several contributions to the code-cache management problem. ...

Keywords: Dynamic optimization, code caches, dynamic translation, just-in-time compilation

12 An LRU-based replacement algorithm augmented with frequency of access in shared chip-multiprocessor caches

Haakon Dybdahl, Per Stenström, Lasse Natvig
September 2006 **Proceedings of the 2006 workshop on MEmory performance: DEaling with Applications, systems and architectures MEDEA '06**
Publisher: ACM Press
Full text available: pdf(105.97 KB) Additional Information: full citation, abstract, references

This paper proposes a new replacement algorithm to protect cache lines with potential future reuse from being evicted. In contrast to the recency based approaches used in the past (LRU for example), our algorithm also uses the notion of *frequency of access*. Instead of evicting the least recently used block, our algorithm identifies among a set of LRU blocks the one that is also least-frequently-used (according to a heuristic) and chooses that as a victim. We have implemented this replacem ...

13 Exploring Code Cache Eviction Granularities in Dynamic Optimization Systems

Kim Hazelwood, James E. Smith
March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '04**
Publisher: IEEE Computer Society
Full text available: pdf(786.32 KB) Additional Information: full citation, abstract, citings, index terms

Dynamic optimization systems store optimized or translatedcode in a software-managed code cache in order tomaximize reuse of transformed code. Code caches storesuperblocks that are not fixed in size, may contain linksto other superblocks, and carry a high replacement overhead.These additional constraints reduce the effectivenessof conventional hardware-based cache management policies.In this paper, we explore code cache managementpolicies that evict large blocks of code from the code cache,thus ...

14 Instrumentation: HDTrans: an open source, low-level dynamic instrumentation system

Swaroop Sridhar, Jonathan S. Shapiro, Eric Northup, Prashanth P. Bungale
June 2006 **Proceedings of the second international conference on Virtual execution environments VEE '06**
Publisher: ACM Press
Full text available: pdf(146.53 KB) Additional Information: full citation, abstract, references, index terms

Dynamic translation is a general purpose tool used for instrumenting programs at run time. Performance of translated execution relies on balancing the cost of translation against the benefits of any optimizations achieved, and many current translators perform substantial rewriting during translation in an attempt to reduce execution time. Our results show that these optimizations offer no significant benefit even when the translated program has a small, hot working set. When used in a broader ra ...

Keywords: binary translation, dynamic instrumentation, dynamic translation

15 Maintaining Consistency and Bounding Capacity of Software Code Caches
Derek Bruening, Saman Amarasinghe
March 2005 **Proceedings of the international symposium on Code generation and optimization CGO '05**
Publisher: IEEE Computer Society
Full text available: pdf(253.56 KB) Additional Information: full citation, abstract, citings, index terms

Software code caches are becoming ubiquitous, in dynamic optimizers, runtime tool platforms, dynamic translators, fast simulators and emulators, and dynamic compilers. Caching frequently executed fragments of code provides significant performance boosts, reducing the overhead of translation and emulation and meeting or exceeding native performance in dynamic optimizers. One disadvantage of caching, memory expansion, can sometimes be ignoredwhen executing a single application. However, as optimiz ...

16 Improving Cost, Performance, and Security of Memory Encryption and Authentication
Chenyu Yan, Daniel Englender, Milos Prvulovic, Brian Rogers, Yan Solihin
May 2006 **ACM SIGARCH Computer Architecture News , Proceedings of the 33rd annual international symposium on Computer Architecture ISCA '06**, Volume 34 Issue 2
Publisher: IEEE Computer Society, ACM Press
Full text available: pdf(363.21 KB) Additional Information: full citation, abstract, index terms

Protection from hardware attacks such as snoopers and mod chips has been receiving increasing attention in computer architecture. This paper presents a new combined memory encryption/authentication scheme. Our new split counters for counter-mode encryption simultaneously eliminate counter overflow problems and reduce per-block counter size, and we also dramatically improve authentication performance and security by using the Galois/Counter Mode of operation (GCM), which leverages counter-mode en ...

17 Code management: Evaluating fragment construction policies for SDT systems
Jason D. Hiser, Daniel Williams, Adrian Filipi, Jack W. Davidson, Bruce R. Childers
June 2006 **Proceedings of the second international conference on Virtual execution environments VEE '06**
Publisher: ACM Press
Full text available: pdf(350.21 KB) Additional Information: full citation, abstract, references, index terms

Software Dynamic Translation (SDT) systems have been used for program instrumentation, dynamic optimization, security policy enforcement, intrusion detection, and many other uses. To be widely applicable, the overhead (runtime, memory usage, and power consumption) should be as low as possible. For instance, if an SDT system is protecting a web server against possible attacks, but causes 30% slowdown, a company

may need 30% more machines to handle the web traffic they expect. Consequently, the ca ...

Keywords: dynamic translation performance, low overhead, performance, software dynamic translator

18 Efficient indexing data structures for flash-based sensor devices

Song Lin, Demetrios Zeinalipour-Yazti, Vana Kalogeraki, Dimitrios Gunopulos, Walid A. Najjar
November 2006 **ACM Transactions on Storage (TOS)**, Volume 2 Issue 4

Publisher: ACM Press

Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, index terms

Flash memory is the most prevalent storage medium found on modern *wireless sensor devices (WSDs)*. In this article we present two external memory index structures for the efficient retrieval of records stored on the local flash memory of a WSD. Our index structures, *MicroHash* and *MicroGF (micro grid files)*, exploit the asymmetric read/write and wear characteristics of flash memory in order to offer high-performance indexing and searching capabilities in the presence of a low- ...

Keywords: Wireless sensor networks, access methods, flash memory

19 Pin: building customized program analysis tools with dynamic instrumentation

Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, Kim Hazelwood
June 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation PLDI '05**, Volume 40 Issue 6

Publisher: ACM Press

Full text available: pdf(300.61 KB) Additional Information: full citation, abstract, references, citings, index terms

Robust and powerful software instrumentation tools are essential for program analysis tasks such as profiling, performance evaluation, and bug detection. To meet this need, we have developed a new instrumentation system called *Pin*. Our goals are to provide *easy-to-use, portable, transparent*, and *efficient* instrumentation. Instrumentation tools (called *Pintools*) are written in C/C++ using Pin's rich API. Pin follows the model of ATOM, allowing the tool writer to analyz ...

Keywords: dynamic compilation, instrumentation, program analysis tools

20 Targeted Path Profiling: Lower Overhead Path Profiling for Staged Dynamic Optimization Systems

Rahul Joshi, Michael D. Bond, Craig Zilles
March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '04**

Publisher: IEEE Computer Society

Full text available: pdf(281.40 KB) Additional Information: full citation, abstract, citings, index terms

In this paper, we present a technique for reducing theoverhead of collecting path profiles in the context of a dynamicoptimizer. The key idea to our approach, called TargetedPath Profiling (TPP), is to use an edge profile to simplifythe collection of a path profile. This notion of profile-guidedprofiling is a natural fit for dynamic optimizers,which typically optimize the code in a series of stages.TPP is an extension to the Ball-Larus Efficient Path Profilingalgorithm. Its increased efficiency ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Terms used **counter cache** and **LRU** and **code cache** and **dynamic code insertion**

Found **198,048** of **198,617**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Managing bounded code caches in dynamic binary optimization systems
Kim Hazelwood, Michael D. Smith
September 2006 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 3 Issue 3
Publisher: ACM Press
Full text available: pdf(666.72 KB) Additional Information: full citation, abstract, references, index terms

Dynamic binary optimizers store altered copies of original program instructions in software-managed code caches in order to maximize reuse of transformed code. Code caches store code blocks that may vary in size, reference other code blocks, and carry a high replacement overhead. These unique constraints reduce the effectiveness of conventional cache management policies. Our work directly addresses these unique constraints and presents several contributions to the code-cache management problem. ...

Keywords: Dynamic optimization, code caches, dynamic translation, just-in-time compilation

2 Exploring Code Cache Eviction Granularities in Dynamic Optimization Systems
Kim Hazelwood, James E. Smith
March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '04**
Publisher: IEEE Computer Society
Full text available: pdf(786.32 KB) Additional Information: full citation, abstract, citings, index terms

Dynamic optimization systems store optimized or translatedcode in a software-managed code cache in order tomaximize reuse of transformed code. Code caches storesuperblocks that are not fixed in size, may contain linksto other superblocks, and carry a high replacement overhead.These additional constraints reduce the effectivenessof conventional hardware-based cache management policies.In this paper, we explore code cache managementpolicies that evict large blocks of code from the code cache,thus ...

3 Generational Cache Management of Code Traces in Dynamic Optimization Systems
Kim Hazelwood, Michael D. Smith
December 2003 **Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture MICRO 36**
Publisher: IEEE Computer Society

Full text available: pdf(393.80 KB) Additional Information: full citation, abstract, citings, index terms

A dynamic optimizer is a runtime software system thatgroups a program's instruction sequences into traces, optimizesthose traces, stores the optimized traces in a software-basedcode cache, and then executes the optimized code inthe code cache. To maximize performance, the vast majorityof the program's execution should occur in the codecache and not in the different aspects of the dynamic optimizationsystem. In the past, designers of dynamic optimizershave used the SPEC2000 benchmark suite to jus ...

4 Maintaining Consistency and Bounding Capacity of Software Code Caches

Derek Bruening, Saman Amarasinghe

March 2005 **Proceedings of the international symposium on Code generation and optimization CGO '05**

Publisher: IEEE Computer Society

Full text available: pdf(253.56 KB) Additional Information: full citation, abstract, citings, index terms

Software code caches are becoming ubiquitous, in dynamic optimizers, runtime tool platforms, dynamic translators, fast simulators and emulators, and dynamic compilers. Caching frequently executed fragments of code provides significant performance boosts, reducing the overhead of translation and emulation and meeting or exceeding native performance in dynamic optimizers. One disadvantage of caching, memory expansion, can sometimes be ignoredwhen executing a single application. However, as optimiz ...

5 The design, implementation, and evaluation of adaptive code unloading for resource-constrained devices

Lingli Zhang, Chandra Krintz

June 2005 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 2 Issue 2

Publisher: ACM Press

Full text available: pdf(814.17 KB) Additional Information: full citation, abstract, references, index terms

Java Virtual Machines (JVMs) for resource-constrained devices, e.g., hand-helds and cell phones, commonly employ interpretation for program translation. However, compilers are able to produce significantly better code quality, and, hence, use device resources more efficiently than interpreters, since compilers can consider large sections of code concurrently and exploit optimization opportunities. Moreover, compilation-based systems store code for reuse by future invocations obviating the redund ...

Keywords: Code unloading, JIT, JVM, code-size reduction, resource-constrained devices

6 Software-based instruction caching for embedded processors

Jason E. Miller, Anant Agarwal

October 2006 **ACM SIGOPS Operating Systems Review , ACM SIGARCH Computer Architecture News , ACM SIGPLAN Notices , Proceedings of the 12th international conference on Architectural support for programming languages and operating systems ASPLOS-XII**, Volume 40 , 34 , 41 Issue 5 , 5 , 11

Publisher: ACM Press

Full text available: pdf(202.35 KB) Additional Information: full citation, abstract, references, index terms

While hardware instruction caches are present in virtually all general-purpose and high-performance microprocessors today, many embedded processors use SRAM or scratchpad memories instead. These are simple array memory structures that are directly addressed and explicitly managed by software. Compared to hardware caches of the same data capacity, they are smaller, have shorter access times and consume less energy per access. Access times are also easier to predict with simple memories since ther ...

Keywords: chaining, instruction cache, software caching

7 Instrumentation: Framework for instruction-level tracing and analysis of program executions

Sanjay Bhansali, Wen-Ke Chen, Stuart de Jong, Andrew Edwards, Ron Murray, Milenko Drinić, Darek Mihočka, Joe Chau

June 2006 **Proceedings of the second international conference on Virtual execution environments VEE '06**

Publisher: ACM Press

Full text available: pdf(227.87 KB) Additional Information: full citation, abstract, references, index terms

Program execution traces provide the most intimate details of a program's dynamic behavior. They can be used for program optimization, failure diagnosis, collecting software metrics like coverage, test prioritization, etc. Two major obstacles to exploiting the full potential of information they provide are: (*i*) performance overhead while collecting traces, and (*ii*) significant size of traces even for short execution scenarios. Reducing information output in an execution trace can r ...

Keywords: callback, code emulation, code replay, time-travel debugging, tracing

8 Compact Binaries with Code Compression in a Software Dynamic Translator

Stacey Shogan, Bruce R. Childers

February 2004 **Proceedings of the conference on Design, automation and test in Europe - Volume 2 DATE '04**

Publisher: IEEE Computer Society

Full text available: pdf(102.73 KB) Additional Information: full citation, abstract, citings, index terms

Embedded software is becoming more flexible and adaptable, which presents new challenges for management of highly constrained system resources. Software dynamic translation (SDT) has been used to enable software malleability at the instruction level for dynamic code optimizers, security checkers, and binary translators. This paper studies the feasibility of using SDT to manage program code storage in embedded systems. We explore to what extent code compression can be incorporated in a software i ...

9 A persistent rescheduled-page cache for low overhead object code compatibility in VLIW architectures

Thomas M. Conte, Sumedh W. Sathaye, Sanjeev Banerjia

December 1996 **Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture MICRO 29**

Publisher: IEEE Computer Society

Full text available: pdf(1.22 MB) Additional Information: full citation, abstract, references, citings, index terms

Object-code compatibility between processor generations is an open issue for VLIW architectures. A potential solution is a technique termed dynamic rescheduling, which performs run-time software rescheduling at the first-time page faults. The time required for rescheduling the pages constitutes a large portion of the overhead of this method. A disk caching scheme that uses a persistent rescheduled-page cache (PRC) is presented. The scheme reduces the overhead associated with dynamic rescheduling ...

Keywords: LRU replacement, VLIW architectures, cache storage, disk caching scheme, dynamic rescheduling, first-time page faults, high-overhead programs, low overhead object code compatibility, operating system support, overhead-based replacement, page replacement policies, persistent rescheduled-page cache, program executions, program performance, run-time software rescheduling, simulations

10 Dynamic translation: Dynamic binary translation for accumulator-oriented architectures

Ho-Seop Kim, James E. Smith

March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '03**

Publisher: IEEE Computer Society

Full text available: pdf(1.13 MB) Additional Information: full citation, abstract, references, citings, index terms

A dynamic binary translation system for a co-designed virtual machine is described and evaluated. The underlying hardware directly executes an accumulator-oriented instruction set that exposes instruction dependence chains (strands) to a distributed microarchitecture containing a simple instruction pipeline. To support conventional program binaries, a source instruction set (Alpha in our study) is dynamically translated to the target accumulator instruction set. The binary translator identifies ...

11 Dynamo: a transparent dynamic optimization system

Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(156.03 KB) Additional Information: full citation, abstract, references, citings, index terms

We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the ...

12 Improving Region Selection in Dynamic Optimization Systems

David Hiniker, Kim Hazelwood, Michael D. Smith

November 2005 **Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture MICRO 38**

Publisher: IEEE Computer Society

Full text available: pdf(432.86 KB) Publisher Site Additional Information: full citation, abstract, citings, index terms

The performance of a dynamic optimization system depends heavily on the code it selects to optimize. Many current systems follow the design of HP Dynamo and select a single interprocedural path, or trace, as the unit of code optimization and code caching. Though this approach to region selection has worked well in practice, we show that it is possible to adapt this basic approach to produce regions with greater locality, less needless code duplication, and fewer profiling counters. In particular ...

13 Wish Branches: Combining Conditional Branching and Predication for Adaptive Predicated Execution

Hyesoon Kim, Onur Mutlu, Jared Stark, Yale N. Patt

November 2005 **Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture MICRO 38**

Publisher: IEEE Computer Society

Full text available: pdf(404.60 KB) Additional Information: full citation, abstract, index terms

Publisher Site

Predicated execution has been used to reduce the number of branch mispredictions by eliminating hard-to-predict branches. However, the additional instruction overhead and additional data dependencies due to predicated execution sometimes offset the performance advantage of having fewer mispredictions. We propose a mechanism in which the compiler generates code that can be executed either as predicated code or non-predicated code (i.e., code with normal conditional branches). The hardware decides ...

14 Compiler Managed Dynamic Instruction Placement in a Low-Power Code Cache

Rajiv A. Ravindran, Pracheeti D. Nagarkar, Ganesh S. Dasika, Eric D. Marsman, Robert M. Senger, Scott A. Mahlke, Richard B. Brown

March 2005 **Proceedings of the international symposium on Code generation and optimization CGO '05**

Publisher: IEEE Computer Society

Full text available: pdf(631.68 KB) Additional Information: full citation, abstract, citings, index terms

Modern embedded microprocessors use low power on-chip memories called scratch-pad memories to store frequently executed instructions and data. Unlike traditional caches, scratch-pad memories lack the complex tag checking and comparison logic, thereby proving to be efficient in area and power. In this work, we focus on exploiting scratch-pad memories for storing hot code segments within an application. Static placement techniques focus on placing the most frequently executed portions of programs ...

15 Cache decay: exploiting generational behavior to reduce cache leakage power

Stefanos Kaxiras, Zhigang Hu, Margaret Martonosi

May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture ISCA '01**, Volume 29 Issue 2

Publisher: ACM Press

Full text available: pdf(1.17 MB) Additional Information: full citation, abstract, references, citings, index terms

Power dissipation is increasingly important in CPUs ranging from those intended for mobile use, all the way up to high-performance processors for high-end servers. While the bulk of the power dissipated is dynamic switching power, leakage power is also beginning to be a concern. Chipmakers expect that in future chip generations, leakage's proportion of total chip power will increase significantly.

This paper examines methods for reducing leakage power within the cache memori ...

16 Trace cache sampling filter

Michael Behar, Avi Mendelson, Avinoam Kolodny

February 2007 **ACM Transactions on Computer Systems (TOCS)**, Volume 25 Issue 1

Publisher: ACM Press

Full text available: pdf(695.83 KB) Additional Information: full citation, abstract, references, index terms

A simple mechanism to increase the utilization of a small trace cache, and simultaneously reduce its power consumption, is presented in this article. The mechanism uses selective storage of traces (filtering) that is based on a new concept in computer architecture: random sampling. The sampling filter exploits the "hot/cold trace" principle, which divides the population of traces into two groups. The first group contains "hot traces" that are executed many times from the ...

Keywords: Trace cache, cache utilization, power dissipation, sampling filter

17 Instrumentation: HDTrans: an open source, low-level dynamic instrumentation system

Swaroop Sridhar, Jonathan S. Shapiro, Eric Northup, Prashanth P. Bungale

June 2006 **Proceedings of the second international conference on Virtual execution environments VEE '06**

Publisher: ACM Press

Full text available: pdf(146.53 KB) Additional Information: full citation, abstract, references, index terms

Dynamic translation is a general purpose tool used for instrumenting programs at run time. Performance of translated execution relies on balancing the cost of translation against the benefits of any optimizations achieved, and many current translators perform substantial rewriting during translation in an attempt to reduce execution time. Our results show that these optimizations offer no significant benefit even when the translated program has a small, hot working set. When used in a broader ra ...

Keywords: binary translation, dynamic instrumentation, dynamic translation

18 Diverge-Merge Processor (DMP): Dynamic Predicated Execution of Complex Control-Flow Graphs Based on Frequently Executed Paths

Hyesoon Kim, Jose A. Joao, Onur Mutlu, Yale N. Patt

December 2006 **Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture MICRO '06**

Publisher: IEEE Computer Society

Full text available: pdf(261.68 KB) Additional Information: full citation, abstract, index terms

This paper proposes a new processor architecture for handling hard-to-predict branches, the diverge-merge processor (DMP). The goal of this paradigm is to eliminate branch mispredictions due to hard-to-predict dynamic branches by dynamically predicating them without requiring ISA support for predicate registers and predicated instructions. To achieve this without incurring large hardware cost and complexity, the compiler provides control-flow information by hints and the processor dynamically pr ...

19 Code management: Dynamic code management: improving whole program code locality in managed runtimes

Xianglong Huang, Brian T Lewis, Kathryn S McKinley

June 2006 **Proceedings of the second international conference on Virtual execution environments VEE '06**

Publisher: ACM Press

Full text available: pdf(153.03 KB) Additional Information: full citation, abstract, references, index terms

Poor code locality degrades application performance by increasing memory stalls due to instruction cache and TLB misses. This problem is particularly an issue for large server applications written in languages such as Java and C# that provide just-in-time (JIT) compilation, dynamic class loading, and dynamic recompilation. However, managed runtimes also offer an opportunity to dynamically profile applications and adapt them to improve their performance. This paper describes a Dynamic Code Manage ...

Keywords: code generation, code layout, dynamic optimization, locality, performance monitoring, virtual machines

20 Instrumentation: Dimension: an instrumentation tool for virtual execution environments

Jing Yang, Shukang Zhou, Mary Lou Soffa

June 2006 **Proceedings of the second international conference on Virtual execution**

environments VEE '06

Publisher: ACM Press

Full text available: pdf(399.54 KB) Additional Information: full citation, abstract, references, index terms

Translation-based virtual execution environments (VEEs) are becoming increasingly popular because of their usefulness. With dynamic translation, a program in a VEE has two binaries: an input source binary and a dynamically generated target binary. Program analysis is important for these binaries, and both the developers and users of VEEs need an instrumentation system to customize program analysis tools. However, existing instrumentation systems for use in VEEs have two drawbacks. First, they ar ...

Keywords: dynamic translation, instrumentation, program analysis tool, virtual execution environment

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next